

日 本 国 特 許 庁
JAPAN PATENT OFFICE

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office

出 願 年 月 日

Date of Application:

2002年 8月19日

出 願 番 号

Application Number:

特願2002-238399

[ST.10/C]:

[JP2002-238399]

出 願 人

Applicant(s):

富士通株式会社

2003年 1月 7日

特 許 庁 長 官
Commissioner,
Japan Patent Office

太田信一郎



出証番号 出証特2002-3103565

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Patent Application of:

Hideo MIYAKE, et al.

Application No.:

Group Art Unit:

Filed: June 25, 2003

Examiner:

For: METHOD OF AND APPARATUS FOR CREATING LOAD MODULE AND COMPUTER
PRODUCT

**SUBMISSION OF CERTIFIED COPY OF PRIOR FOREIGN
APPLICATION IN ACCORDANCE
WITH THE REQUIREMENTS OF 37 C.F.R. § 1.55**

Commissioner for Patents
PO Box 1450
Alexandria, VA 22313-1450

Sir:

In accordance with the provisions of 37 C.F.R. § 1.55, the applicant(s) submit(s)
herewith a certified copy of the following foreign application:

Japanese Patent Application No(s). 2002-187230 and 2002-238399

Filed: June 27, 2002 and August 19, 2002

It is respectfully requested that the applicant(s) be given the benefit of the foreign filing
date(s) as evidenced by the certified papers attached hereto, in accordance with the
requirements of 35 U.S.C. § 119.

Respectfully submitted,

STAAS & HALSEY LLP

Date: June 25, 2003

By: 

H. J. Staas
Registration No. 22,010

1201 New York Ave, N.W., Suite 700
Washington, D.C. 20005
Telephone: (202) 434-1500
Facsimile: (202) 434-1501

【書類名】 特許願

【整理番号】 0240872

【提出日】 平成14年 8月19日

【あて先】 特許庁長官殿

【国際特許分類】 G06F 15/16

【発明の名称】 ロードモジュール生成方法、ロードモジュール生成プログラムおよびロードモジュール生成装置

【請求項の数】 9

【発明者】

【住所又は居所】 神奈川県川崎市中原区上小田中4丁目1番1号 富士通株式会社内

【氏名】 三宅 英雄

【発明者】

【住所又は居所】 神奈川県川崎市中原区上小田中4丁目1番1号 富士通株式会社内

【氏名】 上方 輝彦

【発明者】

【住所又は居所】 神奈川県川崎市中原区上小田中4丁目1番1号 富士通株式会社内

【氏名】 畔上 謙吾

【特許出願人】

【識別番号】 000005223

【氏名又は名称】 富士通株式会社

【代理人】

【識別番号】 100104190

【弁理士】

【氏名又は名称】 酒井 昭徳

【手数料の表示】

【予納台帳番号】 041759

【納付金額】 21,000円

【提出物件の目録】

【物件名】 明細書 1

【物件名】 図面 1

【物件名】 要約書 1

【包括委任状番号】 9906241

【プルーフの要否】 要

【書類名】 明細書

【発明の名称】 ロードモジュール生成方法、ロードモジュール生成プログラム
およびロードモジュール生成装置

【特許請求の範囲】

【請求項 1】 複数のプロセッサを搭載する計算機システムにより実行されるプログラムのロードモジュールを生成するロードモジュール生成方法において、

第 1 のプロセッサにより実行されるプログラムのオブジェクトを結合して所定のメモリ領域を構築する第 1 のメモリ空間構築工程と、

前記第 1 のメモリ空間構築工程で構築された各メモリ領域のアドレスにもとづいて、前記第 1 のプロセッサにより実行されるプログラム中の各シンボルのアドレスを算出する第 1 のメモリ空間内アドレス解決工程と、

第 2 のプロセッサにより実行されるプログラムのオブジェクトを結合して所定のメモリ領域を構築する第 2 のメモリ空間構築工程と、

前記第 2 のメモリ空間構築工程で構築された各メモリ領域のアドレスにもとづいて、前記第 2 のプロセッサにより実行されるプログラム中の各シンボルのアドレスを算出する第 2 のメモリ空間内アドレス解決工程と、

前記第 1 のメモリ空間内アドレス解決工程でアドレスが解決されなかったシンボルのアドレスを、前記第 2 のメモリ空間内アドレス解決工程で算出された当該シンボルのアドレスにもとづいて算出するメモリ空間間アドレス解決工程と、

を含んだことを特徴とするロードモジュール生成方法。

【請求項 2】 前記メモリ空間間アドレス解決工程では、前記第 1 のメモリ空間内アドレス解決工程でアドレスが解決されなかったシンボルのアドレスを、前記第 2 のメモリ空間内アドレス解決工程で算出された当該シンボルのアドレスの、前記第 2 のメモリ空間構築工程で構築された当該シンボルの所属するメモリ領域の開始アドレスからのオフセットにもとづいて算出することを特徴とする請求項 1 に記載のロードモジュール生成方法。

【請求項 3】 前記第 1 のメモリ空間構築工程では、前記第 1 のプロセッサからのみ参照可能なメモリ領域と、前記第 1 のプロセッサおよび前記第 2 のプロ

セッサから参照可能なメモリ領域とを構築することを特徴とする請求項 1 または請求項 2 に記載のロードモジュール生成方法。

【請求項 4】 複数のプロセッサを搭載する計算機システムにより実行されるプログラムのロードモジュールを生成するロードモジュール生成プログラムにおいて、

第 1 のプロセッサにより実行されるプログラムのオブジェクトを結合して所定のメモリ領域を構築する第 1 のメモリ空間構築工程と、

前記第 1 のメモリ空間構築工程で構築された各メモリ領域のアドレスにもとづいて、前記第 1 のプロセッサにより実行されるプログラム中の各シンボルのアドレスを算出する第 1 のメモリ空間内アドレス解決工程と、

第 2 のプロセッサにより実行されるプログラムのオブジェクトを結合して所定のメモリ領域を構築する第 2 のメモリ空間構築工程と、

前記第 2 のメモリ空間構築工程で構築された各メモリ領域のアドレスにもとづいて、前記第 2 のプロセッサにより実行されるプログラム中の各シンボルのアドレスを算出する第 2 のメモリ空間内アドレス解決工程と、

前記第 1 のメモリ空間内アドレス解決工程でアドレスが解決されなかったシンボルのアドレスを、前記第 2 のメモリ空間内アドレス解決工程で算出された当該シンボルのアドレスにもとづいて算出するメモリ空間間アドレス解決工程と、

をコンピュータに実行させることを特徴とするロードモジュール生成プログラム。

【請求項 5】 前記メモリ空間間アドレス解決工程では、前記第 1 のメモリ空間内アドレス解決工程でアドレスが解決されなかったシンボルのアドレスを、前記第 2 のメモリ空間内アドレス解決工程で算出された当該シンボルのアドレスの、前記第 2 のメモリ空間構築工程で構築された当該シンボルの所属するメモリ領域の開始アドレスからのオフセットにもとづいて算出することを特徴とする請求項 4 に記載のロードモジュール生成プログラム。

【請求項 6】 前記第 1 のメモリ空間構築工程では、前記第 1 のプロセッサからのみ参照可能なメモリ領域と、前記第 1 のプロセッサおよび前記第 2 のプロセッサから参照可能なメモリ領域とを構築することを特徴とする請求項 4 または

請求項 5 に記載のロードモジュール生成プログラム。

【請求項 7】 複数のプロセッサを搭載する計算機システムにより実行されるプログラムのロードモジュールを生成するロードモジュール生成装置において

、
第 1 のプロセッサにより実行されるプログラムのオブジェクトを結合して所定のメモリ領域を構築する第 1 のメモリ空間構築手段と、

前記第 1 のメモリ空間構築手段により構築された各メモリ領域のアドレスにもとづいて、前記第 1 のプロセッサにより実行されるプログラム中の各シンボルのアドレスを算出する第 1 のメモリ空間内アドレス解決手段と、

第 2 のプロセッサにより実行されるプログラムのオブジェクトを結合して所定のメモリ領域を構築する第 2 のメモリ空間構築手段と、

前記第 2 のメモリ空間構築手段により構築された各メモリ領域のアドレスにもとづいて、前記第 2 のプロセッサにより実行されるプログラム中の各シンボルのアドレスを算出する第 2 のメモリ空間内アドレス解決手段と、

前記第 1 のメモリ空間内アドレス解決手段によりアドレスが解決されなかったシンボルのアドレスを、前記第 2 のメモリ空間内アドレス解決手段により算出された当該シンボルのアドレスにもとづいて算出するメモリ空間間アドレス解決手段と、

を備えたことを特徴とするロードモジュール生成装置。

【請求項 8】 前記メモリ空間間アドレス解決手段は、前記第 1 のメモリ空間内アドレス解決手段によりアドレスが解決されなかったシンボルのアドレスを、前記第 2 のメモリ空間内アドレス解決手段により算出された当該シンボルのアドレスの、前記第 2 のメモリ空間構築手段により構築された当該シンボルの所属するメモリ領域の開始アドレスからのオフセットにもとづいて算出することを特徴とする請求項 7 に記載のロードモジュール生成装置。

【請求項 9】 前記第 1 のメモリ空間構築手段は、前記第 1 のプロセッサからのみ参照可能なメモリ領域と、前記第 1 のプロセッサおよび前記第 2 のプロセッサから参照可能なメモリ領域とを構築することを特徴とする請求項 7 または請求項 8 に記載のロードモジュール生成装置。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】

この発明は、複数のプロセッサを搭載する計算機システムにより実行されるプログラムのロードモジュールを生成するロードモジュール生成方法、ロードモジュール生成プログラムおよびロードモジュール生成装置に関する。

【0002】

【従来の技術】

近年の計算機システムでは、複数のプロセッサを搭載することでシステムの処理能力を向上させるべく、「分散メモリ型マルチプロセッサ方式 (D i s t r i b u t e d - M e m o r y M u l t i p r o c e s s o r s)」が採用されることがある。

【0003】

図1は、分散メモリ型マルチプロセッサ方式にもとづいた計算機システムを模式的に示す説明図である。図示するように、プロセッサ (P R O C E S S O R) 101とメモリ (M E M O R Y) 102とから構成されるPE (P r o c e s s i n g E l e m e n t : プロセッサ要素) 100がn個、相互接続網 (I N T E R C O N N E C T I O N N E T W O R K) 103により接続されている。

【0004】

また、図2は上記システムにおけるメモリ空間の定義例を模式的に示す説明図である。図示するように個々のプロセッサ101は、同じPE100内のメモリ102だけを読み書きすることができる。

【0005】

そしてこのようなシステムにおいては、MPI (M e s s a g e - P a s s i n g I n t e r f a c e) などのプロセッサ間通信機構を用いることで、SPMD (S i n g l e - P r o g r a m , M u l t i p l e - D a t a) プログラミングにもとづくプログラムが実行されることが多い。

【0006】

図3は上記プログラムの一例を示す説明図である。図示するプログラムはn個

のメモリ 1 0 2 にそれぞれ格納され、 n 個のプロセッサ 1 0 1 によりそれぞれ実行される。プログラムは同一でも、PE 1 0 0 の ID (番号) により処理が分岐するので、 n 個の PE 1 0 0 による並列処理が実現される。

【0 0 0 7】

たとえば図示するプログラムでは、「my__rank」が上記 ID を示す変数であり、my__rank = 0 以外の PE では if 以下の処理が、my__rank = 0 の PE では else 以下の処理が、それぞれ実行されることになる。

【0 0 0 8】

また、図 4 は上記プログラムのロードモジュールの生成手順を示すフローチャートである。まずコンパイラにより、プログラムのソース記述をアセンブリ記述に変換し (ステップ S 4 0 1 ~ S 4 0 3)、次にアセンブラにより、上記アセンブリ記述からオブジェクトを生成する (ステップ S 4 0 4 ~ S 4 0 6)。そして、上記により生成された複数のオブジェクトをリンカにより結合することで (ステップ S 4 0 7 ~ S 4 1 0)、上記プログラムのロードモジュールを生成する。

【0 0 0 9】

ところで、図 1 のような分散メモリ型マルチプロセッサ方式にもとづくシステムは、従来は半導体集積技術の限界から、複数のチップ (および複数のボード) により構成されてきた。しかしながら、近年の半導体集積技術の向上により、複数の PE を一つのチップに収めることが可能となっている。

【0 0 1 0】

この場合、相互接続網を介した PE 間のデータの受け渡しはパケット伝送方式ではなく、共有メモリにデータを直接ストア / 共有メモリからデータを直接ロードすることで、より高速におこなうことができる。このように、複数のプロセッサから読み書きされる共有メモリを設ける方式を、「分散共有メモリ型マルチプロセッサ方式」と呼ぶ。

【0 0 1 1】

図 5 は、分散共有メモリ型マルチプロセッサ方式にもとづいた計算機システムを模式的に示す説明図である。図 1 に示した分散メモリ型マルチプロセッサ方式との差異は、メモリ 5 0 2 に、他の PE 内のプロセッサからも読み書きできる S

M (S h a r e d M e m o r y : 共有メモリ) と、同一の P E 内のプロセッサからしか読み書きできない L M (L o c a l M e m o r y : 固有メモリ) との 2 種類ある点である。

【 0 0 1 2 】

また、図 6 は上記システムにおけるメモリ空間の定義例を示す説明図である。図中、たとえば 1 番の P E (P E # 1) の S M は、0 番の P E (P E # 0) のメモリ空間および 1 番の P E (P E # 1) のメモリ空間に重複して割り当てられている。

【 0 0 1 3 】

仮に、P E # 1 の S M が P E # 0 のメモリ空間では 0×3000 以下、P E # 1 のメモリ空間では 0×2000 以下のアドレスに割り当てられていたとすると、たとえば P E # 0 が 0×2000 にデータを書き込み、P E # 1 が 0×3000 からデータを読み出すことで、P E # 0 と P E # 1 との間で上記データを授受できたことになる。

【 0 0 1 4 】

なお、図示する例では P E # 0 のみが、他のすべての P E の S M を参照・変更することができる。一方 P E # 1 ~ # n の各メモリ空間には、物理的に他の P E に属するメモリが割り当てられていないので、これらの P E は同一 P E 内の L M および S M を参照・変更するのみである。

【 0 0 1 5 】

そして、このような分散共有メモリ型マルチプロセッサ方式の計算機システムでも、分散メモリ型マルチプロセッサ方式と同様、図 3 に示したような S P M D プログラミングにもとづくプログラムを実行することは可能である。

【 0 0 1 6 】

【発明が解決しようとする課題】

しかしながら、分散メモリ型マルチプロセッサ方式にせよ、分散共有メモリ型マルチプロセッサ方式にせよ、個々のプロセッサが実行するのはプログラムの一部（以下では「部分プログラム」という）であるにもかかわらず、各 P E にはプログラムの全体が配分されるので、それだけの容量のメモリを用意しなければな

らず、コストがかさんでしまうという問題があった。

【 0 0 1 7 】

この問題は、少なくとも分散メモリ型マルチプロセッサ方式のシステムについては、プログラムをSPMDでなくMPMD (Multiple-Program, Multiple-Data) プログラミングにもとづいて作成することで回避可能である。

【 0 0 1 8 】

MPMDにもとづくプログラミングでは、SPMDのように各PEにより実行される部分プログラムをすべて結合したようなプログラムでなく、端的にそれぞれのPE向けのプログラムを作成する。図7はPE#0、図8はPE#1～#n向けのプログラムの一例をそれぞれ示す説明図である。図示するように各PE用のプログラムには、他PE用の部分プログラムが含まれないので、その分メモリの容量を小さくすることができる。なお、これらのプログラムのロードモジュールは、図4に示した手順により生成される。

【 0 0 1 9 】

一方、分散共有メモリ型マルチプロセッサ方式では、上述のようにある場所に格納された同一のデータを複数のPEが参照・変更するが、メモリ空間上のそのアドレスはPEごとに異なっている。したがって、各PE向けのプログラムについてリンカでアドレスを解決する際には、対象物が同一でもPEごとに異なるアドレスに変換しなければならないが、従来技術のリンカにはこうした機能がなかった。

【 0 0 2 0 】

そのため分散共有メモリ型マルチプロセッサ方式の計算機システムで動作するプログラムは、SPMDプログラミングによってしか作成することができず、実行時にはスキップされる部分プログラムが各PEに多数配分され、無駄な部分プログラムを保持するために多くのメモリが必要になってしまうという問題があった。

【 0 0 2 1 】

この発明は上記従来技術による問題を解決するため、分散共有メモリ型マルチ

プロセッサ方式を採用する計算機システムにおいても、より少ないメモリ容量で動作するプログラムのロードモジュールを生成することが可能なロードモジュール生成方法、ロードモジュール生成プログラムおよびロードモジュール生成装置を提供することを目的とする。

【 0 0 2 2 】

【課題を解決するための手段】

上述した課題を解決し、目的を達成するため、この発明にかかるロードモジュール生成方法、ロードモジュール生成プログラムまたはロードモジュール生成装置は、複数のプロセッサを搭載する計算機システムにより実行されるプログラムのロードモジュールを生成するロードモジュール生成方法において、第1のプロセッサにより実行されるプログラムおよび第2のプロセッサにより実行されるプログラムについて、それぞれオブジェクトを結合して所定のメモリ領域を構築のうえ各シンボルのアドレスを算出するとともに、第1のプロセッサにより実行されるプログラム中アドレスが解決されなかったシンボルのアドレスを、第2のプロセッサにより実行されるプログラム中での当該シンボルのアドレスにもとづいて算出することを特徴とする。

【 0 0 2 3 】

また、この発明にかかるロードモジュール生成方法、ロードモジュール生成プログラムまたはロードモジュール生成装置は、第1のプロセッサにより実行されるプログラム中アドレスが解決されなかったシンボルのアドレスを、第2のプロセッサにより実行されるプログラム中での当該シンボルのアドレスの、当該シンボルの所属するメモリ領域の開始アドレスからのオフセットにもとづいて算出することを特徴とする。

【 0 0 2 4 】

また、この発明にかかるロードモジュール生成方法、ロードモジュール生成プログラムまたはロードモジュール生成装置は、第1のプロセッサにより実行されるプログラムのオブジェクトを結合して所定のメモリ領域を構築する際、第1のプロセッサからのみ参照可能なメモリ領域と、第1のプロセッサおよび第2のプロセッサから参照可能なメモリ領域とを構築することを特徴とする。

【 0 0 2 5 】

これらの発明によって、MPMDプログラミングにもとづいて作成された各プロセッサ用のソースプログラムから、分散共有メモリ型マルチプロセッサ方式を採用する計算機でも実行可能なそのロードモジュールを生成することができる。逆にいえば、分散共有メモリ型マルチプロセッサ方式の計算機についても、MPMDプログラミングによってプログラムを作成することができる。

【 0 0 2 6 】

【発明の実施の形態】

以下に添付図面を参照して、この発明にかかるロードモジュール生成方法、ロードモジュール生成プログラムおよびロードモジュール生成装置の好適な実施の形態を詳細に説明するが、その前に本発明の基本方針を簡単に説明する。

【 0 0 2 7 】

(基本方針)

図9および図10は、分散共有メモリ型マルチプロセッサ方式の計算機システムを前提とする、MPMDプログラミングにもとづくプログラムの一例を示す説明図である。図9はPE#0用、図10はPE#1用であり、PE#0からPE#1に必要なデータを渡して、所定の処理を依頼した後、その結果を受け取るためのプログラムである。

【 0 0 2 8 】

すなわちまずPE#0において、変数inputを読み出して、その値を変数inに書き込み(図9Th0-1)、次にPE#1の関数Th1の実行を指示する(図9Th0-2)。これを受けたPE#1では、Th1の中で変数inを入力として関数f1を呼び出し、その実行結果を変数outに書き込む(図10Th1-1)。その後、PE#0は変数outを読み出し、その値を変数outputに書き込む(図9Th0-3)。

【 0 0 2 9 】

なお、実際のプログラムではPE#1に処理を依頼した後(すなわちTh0-2の後)、PE#0はPE#1とは無関係な別の処理に移行するが、ここでは簡略化してPE#0-PE#1間の連携部分のみを示している。

【 0 0 3 0 】

上述のように、従来技術の言語処理系ではこれらのソースプログラムについて、実際に実行可能なロードモジュールを生成することができない。たとえば図 9 に示した P E # 0 用のプログラムで、`extern` 宣言されている変数 `i n` および `o u t` は、図 1 0 に示した P E # 1 用のプログラムで定義されているため、P E # 0 用のプログラムをリンクした時点ではアドレスは不定である。

【 0 0 3 1 】

P E # 1 用のプログラムをリンクすると上記変数のアドレスは確定するが、判明するのはあくまで P E # 1 のメモリ空間上でのアドレスであり、当該アドレスで指し示される物理的な記憶領域の、P E # 0 のメモリ空間上におけるアドレスは依然不明である。

【 0 0 3 2 】

そこで、以下に説明する実施の形態のような計算式を用いて、P E # 1 のメモリ空間上でのアドレスから、P E # 0 のメモリ空間上でのアドレスを割り出してやることで、P E # 0 用のプログラムで未解決シンボルとして残った、変数 `i n` および `o u t` のアドレスを解決する。

【 0 0 3 3 】

図 1 1 は、図 9 および図 1 0 に示したプログラムが本発明によりアドレス解決された後の状況を模式的に示す説明図である。同一の変数 `i n / o u t` が、P E # 0 用のプログラムでは `0 x 3 0 0 0 / 0 x 3 0 0 4` に、P E # 1 用のプログラムでは `0 x 2 0 0 0 / 0 x 2 0 0 4` に、それぞれ置換されていることが分かる。

【 0 0 3 4 】

なお、図中「`text area`」とはプログラムの命令列を保持する領域であり、「`data area`」とはプログラムから読み書きされるデータのうち、非共有データすなわち当該プログラムを実行する P E 以外の他の P E からは参照・変更されることのないデータを保持する領域である。これらの領域は物理的には各 P E の L M に配置され、他の P E からの参照や変更はできない。

【 0 0 3 5 】

また、図中「`shared data area # k`」 ($0 \leq k \leq n$) は、

いずれも共有データを保持する領域である。これらの領域は物理的には、それぞれ k 番目の PE の SM に配置され、他の PE から参照・変更される可能性があるものとする。

【 0 0 3 6 】

たとえば変数 i_n の格納場所は、実際には $PE \# 1$ の SM 上の一点であり、同じ場所に $PE \# 0$ は $0 \times 3 0 0 0$ 、 $PE \# 1$ は $0 \times 2 0 0 0$ のアドレスを割り当てているので、いずれの PE からその値を参照・変更することができる。これにより、共有メモリを介した PE 間でのデータの授受が可能となっている。

【 0 0 3 7 】

(実施の形態)

次に図 1 2 は、本発明の実施の形態にかかるロードモジュール生成装置のハードウェア構成の一例を示すブロック図である。

【 0 0 3 8 】

図中、まず $CPU 1 2 0 1$ は装置全体の制御を司る。 $ROM 1 2 0 2$ はブートプログラムなどを記憶している。 $RAM 1 2 0 3$ は $CPU 1 2 0 1$ のワークエリアとして使用される。 $HDD 1 2 0 4$ は、 $CPU 1 2 0 1$ の制御にしたがって $HD 1 2 0 5$ に対するデータのリード／ライトを制御する。 $HD 1 2 0 5$ は、 $HDD 1 2 0 4$ の制御にしたがって書き込まれたデータを記憶する。

【 0 0 3 9 】

$FDD 1 2 0 6$ は、 $CPU 1 2 0 1$ の制御にしたがって $FD 1 2 0 7$ に対するデータのリード／ライトを制御する。 $FD 1 2 0 7$ は、 $FDD 1 2 0 6$ の制御にしたがって書き込まれたデータを記憶したり、記憶しているデータを $FDD 1 2 0 6$ の磁気ヘッドに読み取らせたりする。着脱可能な記録媒体としては、 $FD 1 2 0 7$ のほか $CD-ROM$ 、 $CD-R$ 、 $CD-RW$ 、 MO 、 DVD (Digital Versatile Disk)、メモリカードなどが考えられる。

【 0 0 4 0 】

ディスプレイ $1 2 0 8$ は、たとえば CRT 、 TFT 液晶ディスプレイ、プラズマディスプレイなどであって、カーソルやウィンドウをはじめ、文書、画像などの各種データを表示する。ネットワーク $I/F 1 2 0 9$ は、イーサネット (R)

ケーブル 1 2 1 0 を通じて L A N に接続されるとともに、L A N と装置内部とのデータの送受信を司る。

【 0 0 4 1 】

キーボード 1 2 1 1 は、文字、数値、各種指示などの入力のためのキーを備え、装置内部へのデータの入力をおこなう。タッチパネル式の入力パッドやテンキーなどであってもよい。マウス 1 2 1 2 は、カーソルの移動や範囲選択などをおこなう。ポインティングデバイスとして同様の機能を備えるものであれば、トラックボール、ジョイスティック、十字キー、ジョグダイヤルなどであってもよい。なお、上記各部はバスまたはケーブル 1 2 0 0 により接続されている。

【 0 0 4 2 】

次に、図 1 3 は本発明の実施の形態にかかるロードモジュール生成装置の構成を機能的に示すブロック図である。同図に示す各機能部は、具体的には図 1 2 に示した H D 1 2 0 5、F D 1 2 0 7 などに格納されたプログラム、具体的にはコンパイラ、アセンブラおよびリンカの三つのプログラムを、C P U 1 2 0 1 が R A M 1 2 0 3 に読み出して実行することにより実現される。

【 0 0 4 3 】

図中、1 3 0 0 ～ 1 3 0 2 はコンパイラにより実現され、プログラムのソース記述をアセンブリ記述に変換する機能部である。その詳細な機能は従来技術によるコンパイラと同一である。

【 0 0 4 4 】

すなわち第 1 解析部 1 3 0 0 は、図 4 に示したステップ S 4 0 1 の処理をおこなう機能部であり、指定されたプログラムのソース記述を読み込んで字句解析および構文解析をおこなうとともに、当該プログラムをコンパイラの内部表現へと変換する。

【 0 0 4 5 】

次に、命令列生成部 1 3 0 1 は図 4 に示したステップ S 4 0 2 の処理をおこなう機能部であり、上記内部表現にもとづいてプログラムの動作を実現する命令列を生成するとともに、当該命令列をコンパイラの内部情報に付加する。

【 0 0 4 6 】

次に、アセンブリ記述出力部 1 3 0 2 は図 4 に示したステップ S 4 0 3 の処理をおこなう機能部であり、コンパイラの内部表現および付加されている命令列にもとづいて、上記プログラムのアセンブリ記述を出力する。

【 0 0 4 7 】

また、図 1 3 中 1 3 0 3 ～ 1 3 0 5 はアセンブラにより実現され、コンパイラから出力されたアセンブリ記述をさらにオブジェクトに変換する機能部である。その詳細な機能は従来技術によるアセンブラと同一である。

【 0 0 4 8 】

すなわち第 2 解析部 1 3 0 3 は、図 4 に示したステップ S 4 0 4 の処理をおこなう機能部であり、コンパイラのアセンブリ記述出力部 1 3 0 2 から出力されたアセンブリ記述を読み込んで、字句解析をおこなうとともにアセンブラの内部表現へと変換する。

【 0 0 4 9 】

次に、バイナリ・コード生成部 1 3 0 4 は図 4 に示したステップ S 4 0 5 の処理をおこなう機能部であり、アセンブラの内部表現にもとづいてバイナリ・コード（命令コードを含む）を生成し、当該コードをアセンブラの内部情報に付加する。

【 0 0 5 0 】

次に、オブジェクト出力部 1 3 0 5 は図 4 に示したステップ S 4 0 6 の処理をおこなう機能部であり、アセンブラの内部表現および付加されているバイナリ・コードにもとづいて、上記プログラムのオブジェクトを出力する。

【 0 0 5 1 】

また、図 1 3 中 1 3 0 6 ～ 1 3 1 1 はリンカにより実現され、アセンブラから出力されたオブジェクトを結合して実行可能なロードモジュールを出力する機能部である。1 3 0 7 以下の各部の機能については、後述するフローチャートで説明するが、1 3 0 6 についてのみ先に説明する。

【 0 0 5 2 】

図 1 4 は、メモリ空間定義情報記憶部 1 3 0 6 に保持されるメモリ空間定義情報の内容を模式的に示す説明図である。メモリ空間定義情報とは、上述の「t e

「text area」「data area」などの各メモリ領域を、メモリ空間内のどのアドレスに配置するかをPEごとに定義したものである。

【0053】

図中、たとえばPE#0のメモリ空間上で0x0000から0xffffまでのアドレスが指し示す場所は、物理的にはPE#0のLM上に存在し、ここにPE#0用のプログラムの「text area」が配置される。同様に、PE#1のメモリ空間上で0x0000から0xffffまでのアドレスが指し示す場所は、物理的にはPE#1のLM上に存在し、ここにPE#1用のプログラムの「text area」が配置される。

【0054】

また、PE#0のメモリ空間上で0x3000から0x3fffまでのアドレスが指し示す場所と、PE#1のメモリ空間上で0x2000から0x2fffまでのアドレスが指し示す場所とは同一であり、物理的にはPE#1のSM上に存在する。そして、ここに「shared data area #1」、すなわちPE#0からも参照・変更可能なPE#1の共有データが配置される。

【0055】

次に、図15は本発明の実施の形態にかかるロードモジュール生成装置における、図9および図10に示したプログラムのロードモジュール生成処理の手順を示すフローチャートである。もっとも、コンパイラおよびアセンブラによる処理は従来技術と同様であるので、同図にはリンカによる処理の手順のみを示している。

【0056】

まず、リンカにより実現されるオブジェクト読み込み部1307が、アセンブラのオブジェクト出力部1305から出力されたオブジェクトのうち、k番目（ $0 \leq k \leq n$ ）のPE用のものをリンカの内部表現として読み込む（ステップS1501）。

【0057】

次にメモリ空間構築部1308が、リンカの内部表現において、k番目のPEの各メモリ領域（上述の「text area」や「data area」など

）を形成し、リンカの内部表現として付加する（ステップ S1502）。

【0058】

次にメモリ空間内アドレス解決部 1309 が、リンカの内部表現において、 k 番目の PE のメモリ空間内の各メモリ領域のアドレス解決をおこなう（ステップ S1503）。そして、ここまでの処理を PE # 0 ($k=0$) から PE # n ($k=n$) のすべてについておこなう。

【0059】

次にメモリ空間間アドレス解決部 1310 は、上記の処理で得られた各 PE のメモリ空間イメージと、図 14 に示したメモリ空間定義情報とを参照して、ステップ S1503 による各メモリ空間内でのアドレス解決では未解決のまま残ったシンボルについて、メモリ空間をまたがったアドレス解決をおこなう（ステップ S1504）。

【0060】

メモリ空間をまたがったアドレス解決とは、具体的にはたとえば図 9 に示した PE # 0 用のプログラムにおいて、PE # 1 用のプログラムで宣言されているためにアドレスが解決できない変数 in および out について、当該変数の PE # 1 のメモリ空間上でのアドレスから、PE # 0 のメモリ空間上でのアドレスを算出する処理である。

【0061】

あるシンボルのある PE におけるアドレスを、他 PE におけるアドレスから算出するための計算式は下記の通りである。

$symbol\ address = self\ base\ address + offset$

ただし $offset = other's\ PE\ symbol\ address - other's\ PE\ base\ address$

【0062】

たとえば変数 out の PE # 0 におけるアドレスは、 $offset = 4$ ($= 0 \times 2004 - 0 \times 2000$) であることから、 $symbol\ address = 0 \times 3004$ ($= 0 \times 3000 + 0 \times 0004$) となる。すなわち図 14 のメモ

リ空間定義情報から、少なくともPE # 0のメモリ空間上でPE # 1との共有データが配置される開始アドレスは分かるので、変数outのオフセットを当該開始アドレスに足し合わせることで、PE # 0にとっての当該変数のアドレスを割り出している。

【0063】

メモリ空間内アドレス解決部1309およびメモリ空間間アドレス解決部1310によるアドレス解決の後、未解決シンボルは存在しないはずなので、次にロードモジュール出力部1311がリンカの内部表現にもとづいて、k番目のPE用のプログラムのロードモジュールを出力する（ステップS1505）。そして、PE # 0（k=0）からPE # n（k=n）のすべてについて同様にロードモジュールを出力した時点で、全PEについてソースからロードモジュールまでのプログラムの変換処理が終了する。

【0064】

なお、図6に示したメモリ空間定義例のもとでは、物理的に他PEのメモリに存在する変数を参照・変更するプログラムはPE # 0用のものに限定される（PE # 1～# nの各メモリ空間には他PEのメモリが割り当てられていないため、これらのPEについて、他PEの変数を参照・変更するようなプログラムは作成することができない）。したがって、PE # 1～# n用のプログラムのリンク時には、ステップS1503の後未解決シンボルは残らないはずなので、ステップS1504によるアドレス解決は不要である。

【0065】

もっとも、図6のようにPE # 0のみが他のPEのメモリを読み書きできるようにする必然性はなく（図6のような定義は一例に過ぎない）、PE # 0以外のPEからも他PEのメモリを参照・変更できるようにしてもよい。

【0066】

その場合PE # 0以外でも、ステップS1503の処理だけでは解決できないシンボルが発生しうるため、たとえば図16に示すように、PE # 1～# nについてもステップS1504によるアドレス解決をおこなう。PE # 0について行ったのと同様の処理を他PEについてもおこなうというのみで、図15の手順と

処理の内容が異なるわけではない。逆に図 1 5 の手順は、図 1 6 において P E # 0 以外の P E につきステップ S 1 5 0 4 を省略できる特別の場合であるとみることとできる。

【 0 0 6 7 】

以上説明したように、本発明によれば分散共有メモリ型マルチプロセッサ方式に対応した、MPMDプログラミングにもとづくプログラムのロードモジュールを生成することが可能である。逆にいえば本発明により、分散共有メモリ型マルチプロセッサ方式を前提とするプログラムを、MPMDプログラミングにより作成することができるので、各 P E にはそこで実行される部分プログラムのみを配置することができ、チップに搭載するメモリの容量を少なくすることができる。

【 0 0 6 8 】

なお、本実施の形態におけるロードモジュール生成方法は、あらかじめ用意されたプログラム（コンパイラ、アセンブラおよびリンカ）がパーソナルコンピュータ、ワークステーションなどの各種のコンピュータ上で実行されることにより実現されるが、このプログラムは H D 、 F D 、 C D - R O M 、 M O 、 D V D などのコンピュータで読み取り可能な各種の記録媒体に記録され、当該記録媒体によって配布することができるほか、インターネットなどのネットワークを介して配布することも可能である。

【 0 0 6 9 】

（付記 1）複数のプロセッサを搭載する計算機システムにより実行されるプログラムのロードモジュールを生成するロードモジュール生成方法において、

第 1 のプロセッサにより実行されるプログラムのオブジェクトを結合して所定のメモリ領域を構築する第 1 のメモリ空間構築工程と、

前記第 1 のメモリ空間構築工程で構築された各メモリ領域のアドレスにもとづいて、前記第 1 のプロセッサにより実行されるプログラム中の各シンボルのアドレスを算出する第 1 のメモリ空間内アドレス解決工程と、

第 2 のプロセッサにより実行されるプログラムのオブジェクトを結合して所定のメモリ領域を構築する第 2 のメモリ空間構築工程と、

前記第 2 のメモリ空間構築工程で構築された各メモリ領域のアドレスにもとづ

いて、前記第 2 のプロセッサにより実行されるプログラム中の各シンボルのアドレスを算出する第 2 のメモリ空間内アドレス解決工程と、

前記第 1 のメモリ空間内アドレス解決工程でアドレスが解決されなかったシンボルのアドレスを、前記第 2 のメモリ空間内アドレス解決工程で算出された当該シンボルのアドレスにもとづいて算出するメモリ空間間アドレス解決工程と、

を含んだことを特徴とするロードモジュール生成方法。

【 0 0 7 0 】

（付記 2）前記メモリ空間間アドレス解決工程では、前記第 1 のメモリ空間内アドレス解決工程でアドレスが解決されなかったシンボルのアドレスを、前記第 2 のメモリ空間内アドレス解決工程で算出された当該シンボルのアドレスの、前記第 2 のメモリ空間構築工程で構築された当該シンボルの所属するメモリ領域の開始アドレスからのオフセットにもとづいて算出することを特徴とする付記 1 に記載のロードモジュール生成方法。

【 0 0 7 1 】

（付記 3）前記第 1 のメモリ空間構築工程では、前記第 1 のプロセッサからのみ参照可能なメモリ領域と、前記第 1 のプロセッサおよび前記第 2 のプロセッサから参照可能なメモリ領域とを構築することを特徴とする付記 1 または付記 2 に記載のロードモジュール生成方法。

【 0 0 7 2 】

（付記 4）前記第 1 のプロセッサおよび前記第 2 のプロセッサから参照可能なメモリ領域は、前記第 2 のプロセッサと同一のプロセッサ要素内のメモリに配置されることを特徴とする付記 3 に記載のロードモジュール生成方法。

【 0 0 7 3 】

（付記 5）複数のプロセッサを搭載する計算機システムにより実行されるプログラムのロードモジュールを生成するロードモジュール生成プログラムにおいて、

第 1 のプロセッサにより実行されるプログラムのオブジェクトを結合して所定のメモリ領域を構築する第 1 のメモリ空間構築工程と、

前記第 1 のメモリ空間構築工程で構築された各メモリ領域のアドレスにもとづいて、前記第 1 のプロセッサにより実行されるプログラム中の各シンボルのアド

レスを算出する第 1 のメモリ空間内アドレス解決工程と、

第 2 のプロセッサにより実行されるプログラムのオブジェクトを結合して所定のメモリ領域を構築する第 2 のメモリ空間構築工程と、

前記第 2 のメモリ空間構築工程で構築された各メモリ領域のアドレスにもとづいて、前記第 2 のプロセッサにより実行されるプログラム中の各シンボルのアドレスを算出する第 2 のメモリ空間内アドレス解決工程と、

前記第 1 のメモリ空間内アドレス解決工程でアドレスが解決されなかったシンボルのアドレスを、前記第 2 のメモリ空間内アドレス解決工程で算出された当該シンボルのアドレスにもとづいて算出するメモリ空間間アドレス解決工程と、

をコンピュータに実行させることを特徴とするロードモジュール生成プログラム。

【 0 0 7 4 】

（付記 6）前記メモリ空間間アドレス解決工程では、前記第 1 のメモリ空間内アドレス解決工程でアドレスが解決されなかったシンボルのアドレスを、前記第 2 のメモリ空間内アドレス解決工程で算出された当該シンボルのアドレスの、前記第 2 のメモリ空間構築工程で構築された当該シンボルの所属するメモリ領域の開始アドレスからのオフセットにもとづいて算出することを特徴とする付記 5 に記載のロードモジュール生成プログラム。

【 0 0 7 5 】

（付記 7）前記第 1 のメモリ空間構築工程では、前記第 1 のプロセッサからのみ参照可能なメモリ領域と、前記第 1 のプロセッサおよび前記第 2 のプロセッサから参照可能なメモリ領域とを構築することを特徴とする付記 5 または付記 6 に記載のロードモジュール生成プログラム。

【 0 0 7 6 】

（付記 8）前記第 1 のプロセッサおよび前記第 2 のプロセッサから参照可能なメモリ領域は、前記第 2 のプロセッサと同一のプロセッサ要素内のメモリに配置されることを特徴とする付記 7 に記載のロードモジュール生成プログラム。

【 0 0 7 7 】

（付記 9）複数のプロセッサを搭載する計算機システムにより実行されるプロゲ

ラムのロードモジュールを生成するロードモジュール生成装置において、

第 1 のプロセッサにより実行されるプログラムのオブジェクトを結合して所定のメモリ領域を構築する第 1 のメモリ空間構築手段と、

前記第 1 のメモリ空間構築手段により構築された各メモリ領域のアドレスにもとづいて、前記第 1 のプロセッサにより実行されるプログラム中の各シンボルのアドレスを算出する第 1 のメモリ空間内アドレス解決手段と、

第 2 のプロセッサにより実行されるプログラムのオブジェクトを結合して所定のメモリ領域を構築する第 2 のメモリ空間構築手段と、

前記第 2 のメモリ空間構築手段により構築された各メモリ領域のアドレスにもとづいて、前記第 2 のプロセッサにより実行されるプログラム中の各シンボルのアドレスを算出する第 2 のメモリ空間内アドレス解決手段と、

前記第 1 のメモリ空間内アドレス解決手段によりアドレスが解決されなかったシンボルのアドレスを、前記第 2 のメモリ空間内アドレス解決手段により算出された当該シンボルのアドレスにもとづいて算出するメモリ空間間アドレス解決手段と、

を備えたことを特徴とするロードモジュール生成装置。

【 0 0 7 8 】

(付記 1 0) 前記メモリ空間間アドレス解決手段は、前記第 1 のメモリ空間内アドレス解決手段によりアドレスが解決されなかったシンボルのアドレスを、前記第 2 のメモリ空間内アドレス解決手段により算出された当該シンボルのアドレスの、前記第 2 のメモリ空間構築手段により構築された当該シンボルの所属するメモリ領域の開始アドレスからのオフセットにもとづいて算出することを特徴とする付記 9 に記載のロードモジュール生成装置。

【 0 0 7 9 】

(付記 1 1) 前記第 1 のメモリ空間構築手段は、前記第 1 のプロセッサからのみ参照可能なメモリ領域と、前記第 1 のプロセッサおよび前記第 2 のプロセッサから参照可能なメモリ領域とを構築することを特徴とする付記 9 または付記 1 0 に記載のロードモジュール生成装置。

【 0 0 8 0 】

(付記 1 2) 前記第 1 のプロセッサおよび前記第 2 のプロセッサから参照可能なメモリ領域は、前記第 2 のプロセッサと同一のプロセッサ要素内のメモリに配置されることを特徴とする付記 1 1 に記載のロードモジュール生成装置。

【 0 0 8 1 】

【発明の効果】

以上説明したように本発明によれば、MPMDプログラミングにもとづいて作成された各プロセッサ用のソースプログラムから、分散共有メモリ型マルチプロセッサ方式を採用する計算機でも実行可能なそのロードモジュールを生成することができるので（逆にいえば、分散共有メモリ型マルチプロセッサ方式の計算機についても、MPMDプログラミングによってプログラムを作成することができるので）、これによって、分散共有メモリ型マルチプロセッサ方式を採用する計算機システムにおいても、より少ないメモリ容量で動作するプログラムのロードモジュールを生成することが可能なロードモジュール生成方法、ロードモジュール生成プログラムおよびロードモジュール生成装置が得られるという効果を奏する。

【図面の簡単な説明】

【図 1】

分散メモリ型マルチプロセッサ方式にもとづいた計算機システムを模式的に示す説明図である。

【図 2】

分散メモリ型マルチプロセッサ方式にもとづいた計算機システムにおける、メモリ空間の定義例を模式的に示す説明図である。

【図 3】

分散メモリ型マルチプロセッサ方式にもとづいた計算機システムで実行される、SPMDプログラミングにもとづくプログラムの一例を示す説明図である。

【図 4】

図 3 に示したプログラムのロードモジュールの生成手順を示すフローチャートである。

【図 5】

分散共有メモリ型マルチプロセッサ方式にもとづいた計算機システムを模式的に示す説明図である。

【図 6】

分散共有メモリ型マルチプロセッサ方式にもとづいた計算機システムにおける、メモリ空間の定義例を模式的に示す説明図である。

【図 7】

分散メモリ型マルチプロセッサ方式にもとづいた計算機システムで実行される、MPMDプログラミングにもとづくプログラムの一例（PE # 0 用）を示す説明図である。

【図 8】

分散メモリ型マルチプロセッサ方式にもとづいた計算機システムで実行される、MPMDプログラミングにもとづくプログラムの一例（PE # 1 ～ # n 用）を示す説明図である。

【図 9】

分散共有メモリ型マルチプロセッサ方式にもとづいた計算機システムで実行される、MPMDプログラミングにもとづくプログラムの一例（PE # 0 用）を示す説明図である。

【図 1 0】

分散共有メモリ型マルチプロセッサ方式にもとづいた計算機システムで実行される、MPMDプログラミングにもとづくプログラムの一例（PE # 1 用）を示す説明図である。

【図 1 1】

図 9 および図 1 0 に示したプログラムが本発明によりアドレス解決された後の状況を模式的に示す説明図である。

【図 1 2】

本発明の実施の形態にかかるロードモジュール生成装置のハードウェア構成の一例を示すブロック図である。

【図 1 3】

本発明の実施の形態にかかるロードモジュール生成装置の構成を機能的に示す

ブロック図である。

【図 1 4】

メモリ空間定義情報記憶部 1 3 0 6 に保持されるメモリ空間定義情報の内容を模式的に示す説明図である。

【図 1 5】

本発明の実施の形態にかかるロードモジュール生成装置における、図 9 および図 1 0 に示したプログラムのロードモジュール生成処理の手順を示すフローチャートである。

【図 1 6】

本発明の実施の形態にかかるロードモジュール生成装置における、図 9 および図 1 0 に示したプログラムのロードモジュール生成処理の手順を示すフローチャートである。

【符号の説明】

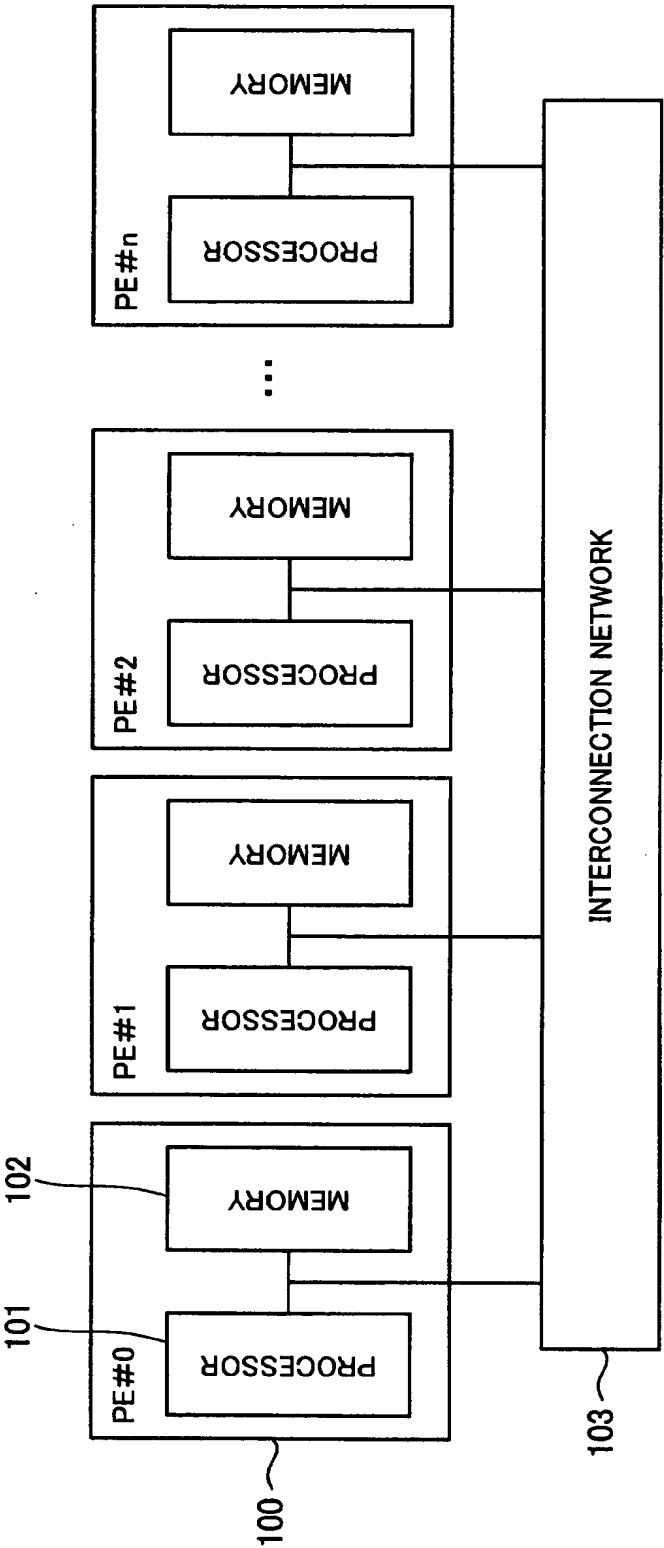
- 1 2 0 0 バスまたはケーブル
- 1 2 0 1 CPU
- 1 2 0 2 ROM
- 1 2 0 3 RAM
- 1 2 0 4 HDD
- 1 2 0 5 HD
- 1 2 0 6 FDD
- 1 2 0 7 FD
- 1 2 0 8 ディスプレイ
- 1 2 0 9 ネットワーク I / F
- 1 2 1 0 イーサネット (R) ケーブル
- 1 2 1 1 キーボード
- 1 2 1 2 マウス
- 1 3 0 0 第 1 解析部
- 1 3 0 1 命令列生成部
- 1 3 0 2 アセンブリ記述出力部

- 1 3 0 3 第 2 解析部
- 1 3 0 4 バイナリ・コード生成部
- 1 3 0 5 オブジェクト出力部
- 1 3 0 6 メモリ空間定義情報記憶部
- 1 3 0 7 オブジェクト読み込み部
- 1 3 0 8 メモリ空間構築部
- 1 3 0 9 メモリ空間内アドレス解決部
- 1 3 1 0 メモリ空間間アドレス解決部
- 1 3 1 1 ロードモジュール出力部

【書類名】 図面

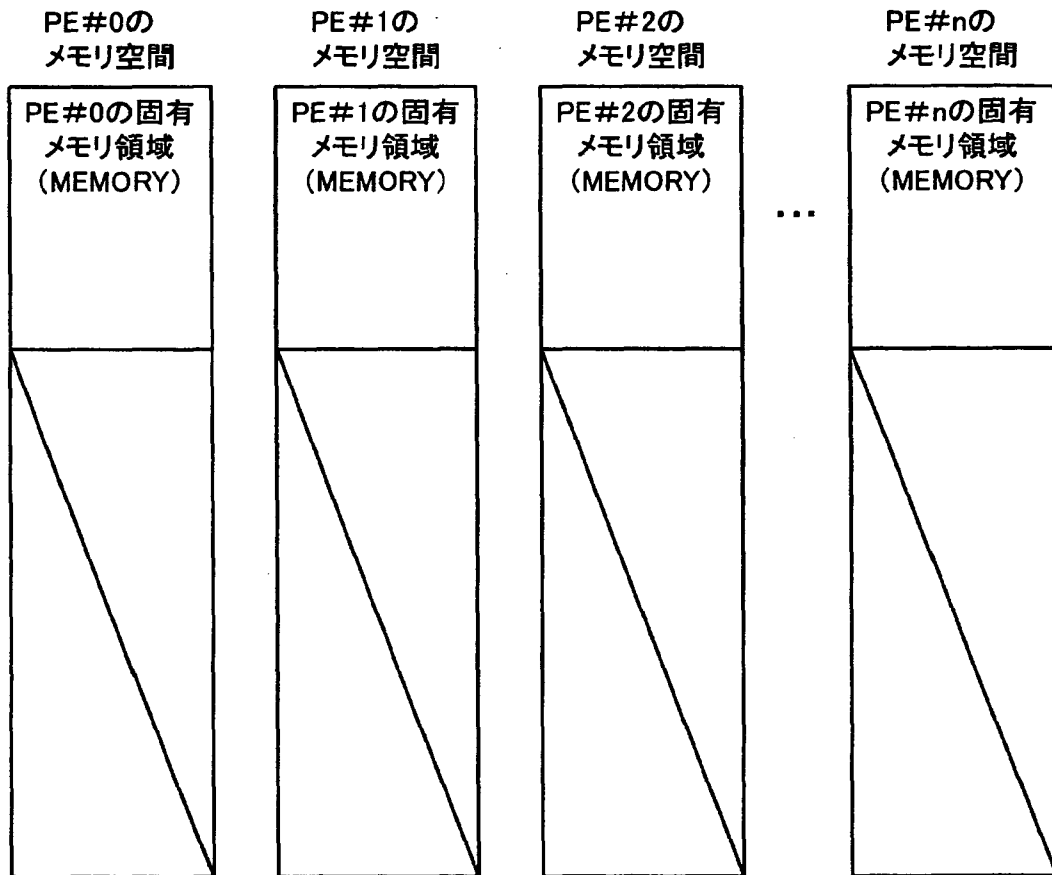
【図 1】

分散メモリ型マルチプロセッサ方式にもとづいた計算機システムを模式的に示す説明図



【図 2】

分散メモリ型マルチプロセッサ方式にもとづいた計算機システムにおける、
メモリ空間の定義例を模式的に示す説明図



【図 3】

分散メモリ型マルチプロセッサ方式にもとづいた計算機システムで実行される、
SPMDプログラミングにもとづくプログラムの一例を示す説明図

```
#include <stdio.h>
#include <string.h>
#include "mpi.h"

int
main(int argc, char **argv)
{
    int my_rank; /* カレントプロセスのランク */
    int source; /* 送信プロセスのランク */
    int dest; /* 受信プロセスのランク */
    int tag=0; /* メッセージのタグ */
    char message[100]; /* メッセージの保管場所 */
    MPI_Status status; /* 受信の戻りステータス */

    /* MPI のスタートアップ */
    MPI_Init(&argc, argv);

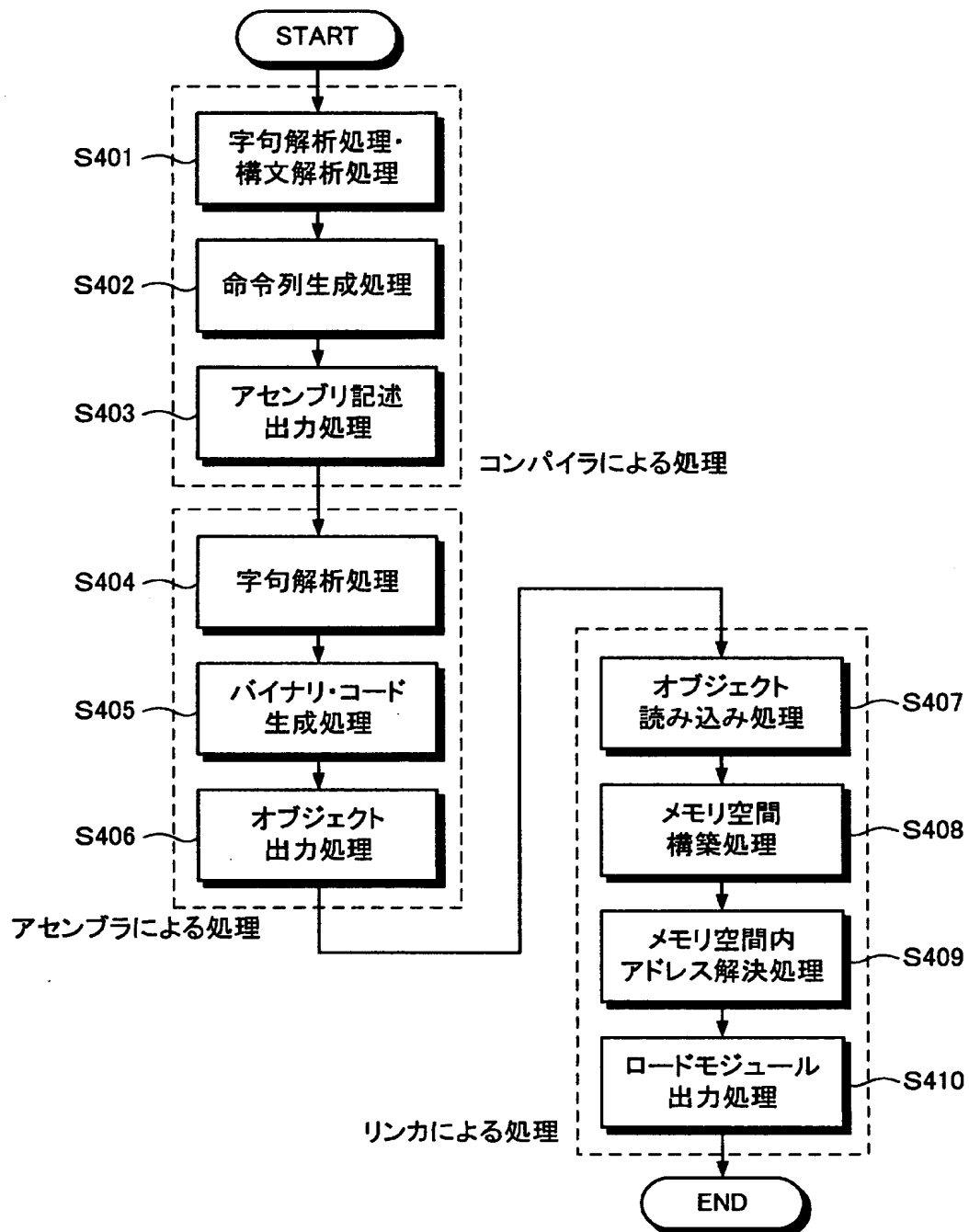
    /* カレントプロセスのランクを求める */
    MPI_Comm_rank(MPI_COMM_WORLD, &my_rank);

    if (my_rank != 0) {
        /* メッセージの生成 */
        sprintf(message, "Greetings from process %d\n", my_rank);
        dest=0;
        /* '\0' も送信するので strlen+1 を使う */
        MPI_Send(message, strlen(message)+1, MPI_CHAR, dest, tag,
        MPI_COMM_WORLD);
    } else {
        source=1;
        MPI_Recv(message, sizeof(message), MPI_CHAR, source, tag,
        MPI_COMM_WORLD, &status);
        printf("%s\n", message);
    }

    /* MPI のシャットダウン */
    MPI_Finalize();
    return 0;
}
```

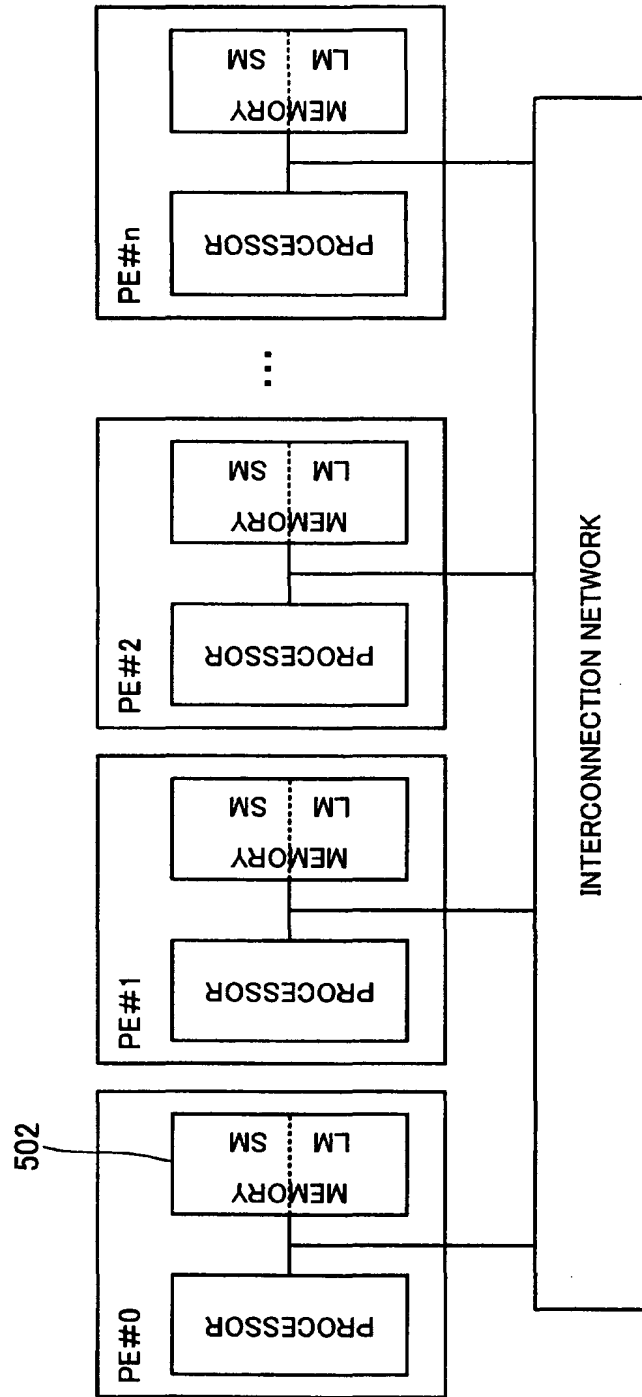
【図 4】

図3に示したプログラムのロードモジュールの生成手順を示すフローチャート



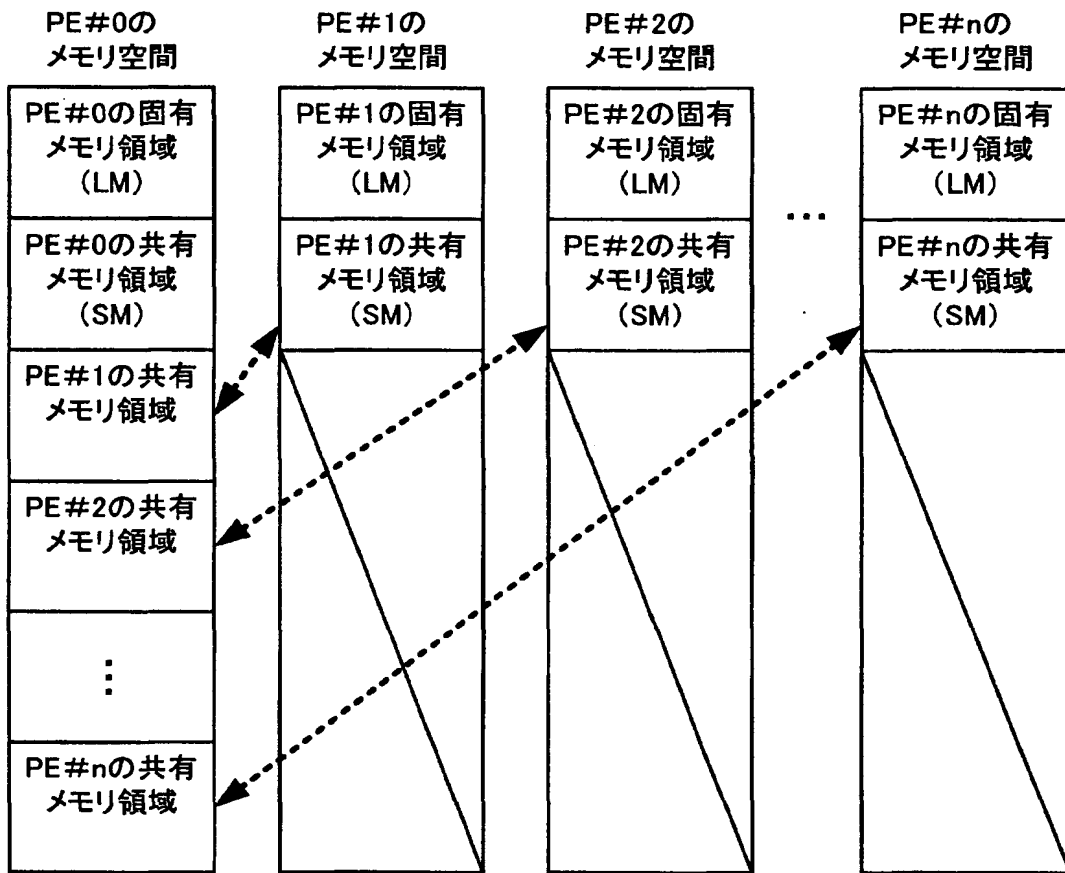
【図 5】

分散共有メモリ型マルチプロセッサ方式にもとづいた計算機システムを模式的に示す説明図



【図 6】

分散共有メモリ型マルチプロセッサ方式にもとづいた計算機システムにおける、
メモリ空間の定義例を模式的に示す説明図



【図 7】

分散メモリ型マルチプロセッサ方式にもとづいた計算機システムで実行される、
MPMDプログラミングにもとづくプログラムの一例(PE#0用)を示す説明図

```
#include <stdio.h>
#include <string.h>
#include "mpi.h"

int
main(int argc, char **argv)
{
    int my_rank; /* カレントプロセスのランク */
    int source; /* 送信プロセスのランク */
    int tag=0; /* メッセージのタグ */
    char message[100]; /* メッセージの保管場所 */
    MPI_Status status; /* 受信の戻りステータス */

    /* MPI のスタートアップ */
    MPI_Init(&argc, argv);

    /* カレントプロセスのランクを求める */
    MPI_Comm_rank(MPI_COMM_WORLD, &my_rank);

    source=1;
    MPI_Recv(message, sizeof(message), MPI_CHAR, source, tag,
    MPI_COMM_WORLD, &status);
    printf("%s\n", message);

    /* MPI のシャットダウン */
    MPI_Finalize();
    return 0;
}
```

【図 8】

分散メモリ型マルチプロセッサ方式にもとづいた計算機システムで実行される、
MPMDプログラミングにもとづくプログラムの一例(PE#1～#n用)を示す説明図

```
#include <stdio.h>
#include <string.h>
#include "mpi.h"

int
main(int argc, char **argv)
{
    int my_rank; /* カレントプロセスのランク */
    int dest; /* 受信プロセスのランク */
    int tag=0; /* メッセージのタグ */
    char message[100]; /* メッセージの保管場所 */

    /* MPI のスタートアップ */
    MPI_Init(&argc, argv);

    /* カレントプロセスのランクを求める */
    MPI_Comm_rank(MPI_COMM_WORLD, &my_rank);

    /* メッセージの生成 */
    sprintf(message, "Greetings from process %d\n", my_rank);
    dest=0;
    /* '\0' も送信するので strlen+1 を使う */
    MPI_Send(message, strlen(message)+1, MPI_CHAR, dest, tag,
             MPI_COMM_WORLD);

    /* MPI のシャットダウン */
    MPI_Finalize();
    return 0;
}
```

【図 9】

分散共有メモリ型マルチプロセッサ方式にもとづいた計算機システムで実行される、
MPMDプログラミングにもとづくプログラムの一例(PE#0用)を示す説明図

```

int input;
int output;
extern int in;
extern int out;

void
Th0(void)
{
    MOVE(&in, &input, sizeof(in));      /* Th0-1 */
    START(1."Th1");                      /* Th0-2 */
    MOVE(&output, &out, sizeof(output)); /* Th0-3 */
}

```

【図 1 0】

分散共有メモリ型マルチプロセッサ方式にもとづいた計算機システムで実行される、
MPMDプログラミングにもとづくプログラムの一例(PE#1用)を示す説明図

```

int in;
int out;

void
Th1(void)
{
    extern void f1(int *, int *);

    f1(&in, &out);                      /* Th1-1 */
}

```

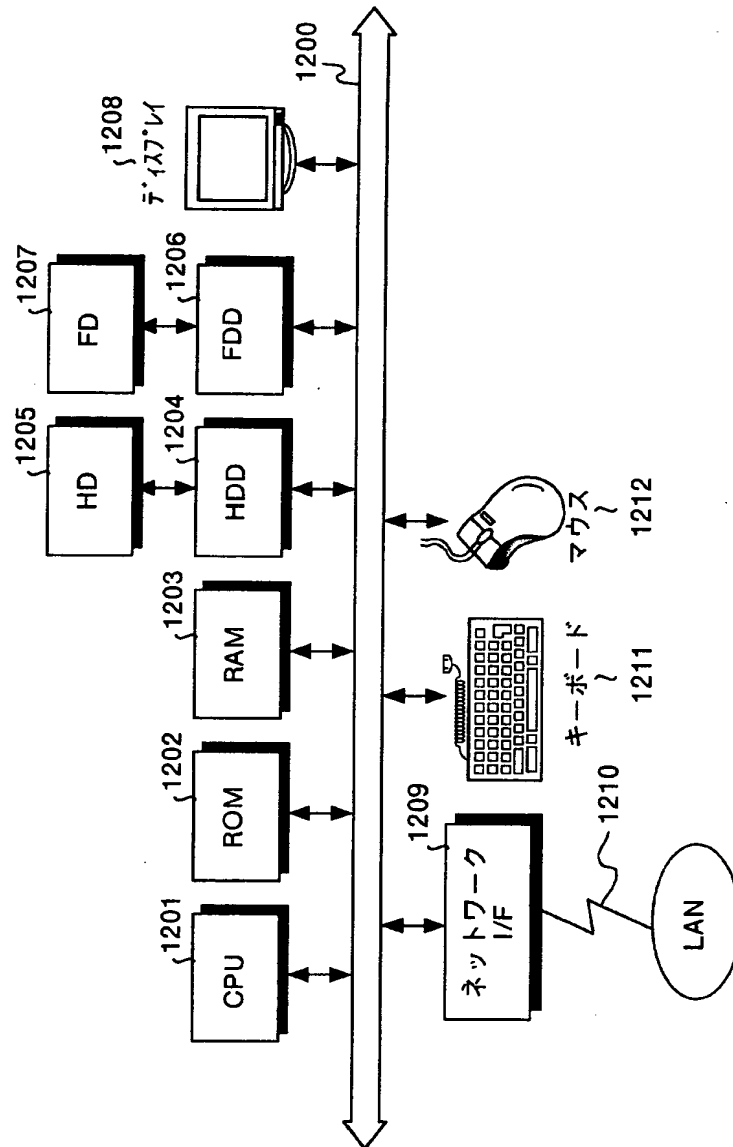
【図 1 1】

図9および図10に示したプログラムが本発明によりアドレス解決された後の
状況を模式的に示す説明図

	PE #0のメモリ空間		PE #1のメモリ空間	
	アドレス	内容	アドレス	内容
text area	0x0000	void Th0(void) { MOVE(0x3000, 0x1000, sizeof(in)); START(1, "Th1"); MOVE(0x1004, 0x3004, sizeof(output)); }	0x0000	void Th1(void) { f1(0x2000, 0x2004); }
data area	0x1000	int input;	0x1000	
	0x1004	int output;		
shared data area #0	0x2000			
shared data area #1	0x3000	int in;	0x2000	int in;
	0x3004	int out;	0x2004	int out;

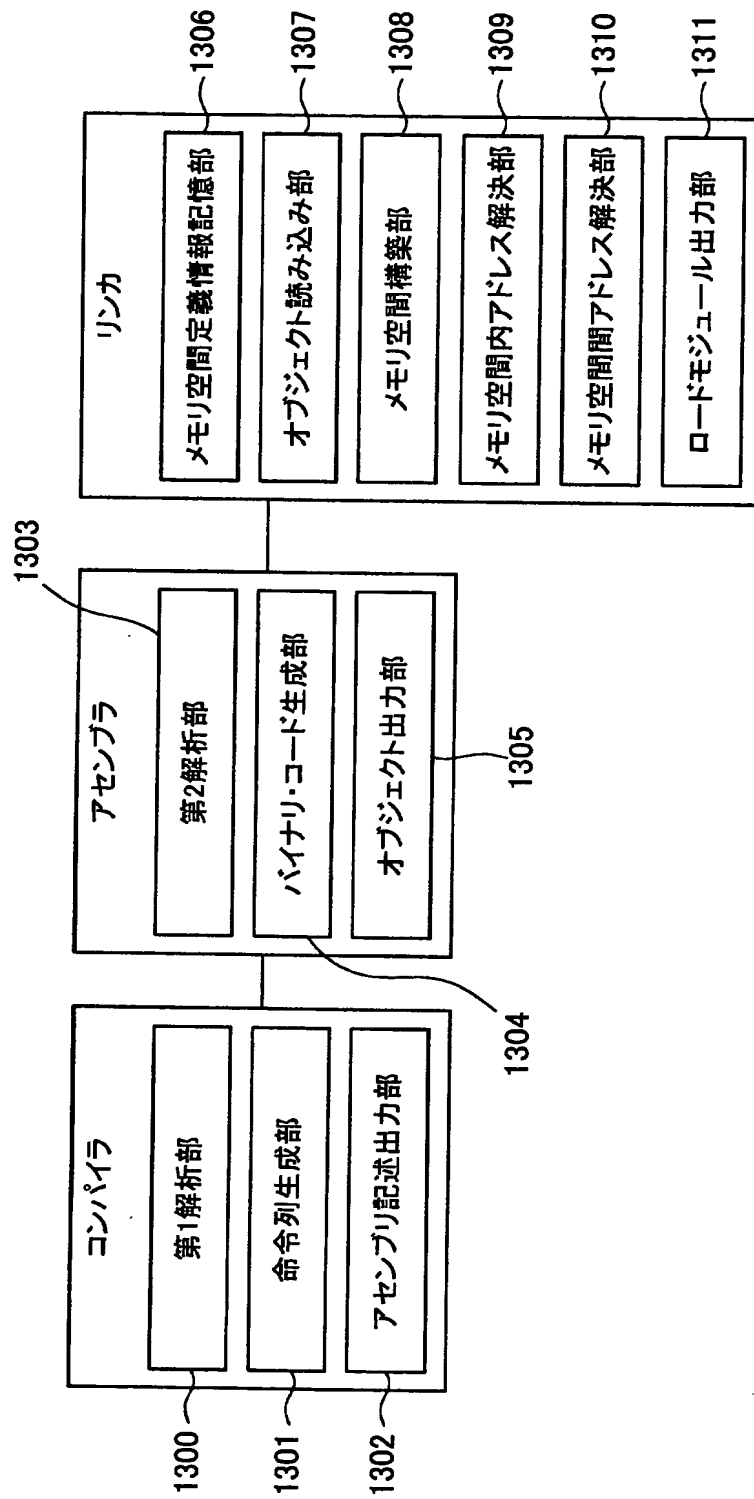
【図 1 2】

本発明の実施の形態にかかるロードモジュール生成装置の
ハードウェア構成の一例を示すブロック図



【図 1 3】

本発明の実施の形態にかかるロードモジュール生成装置の構成を機能的に示すブロック図



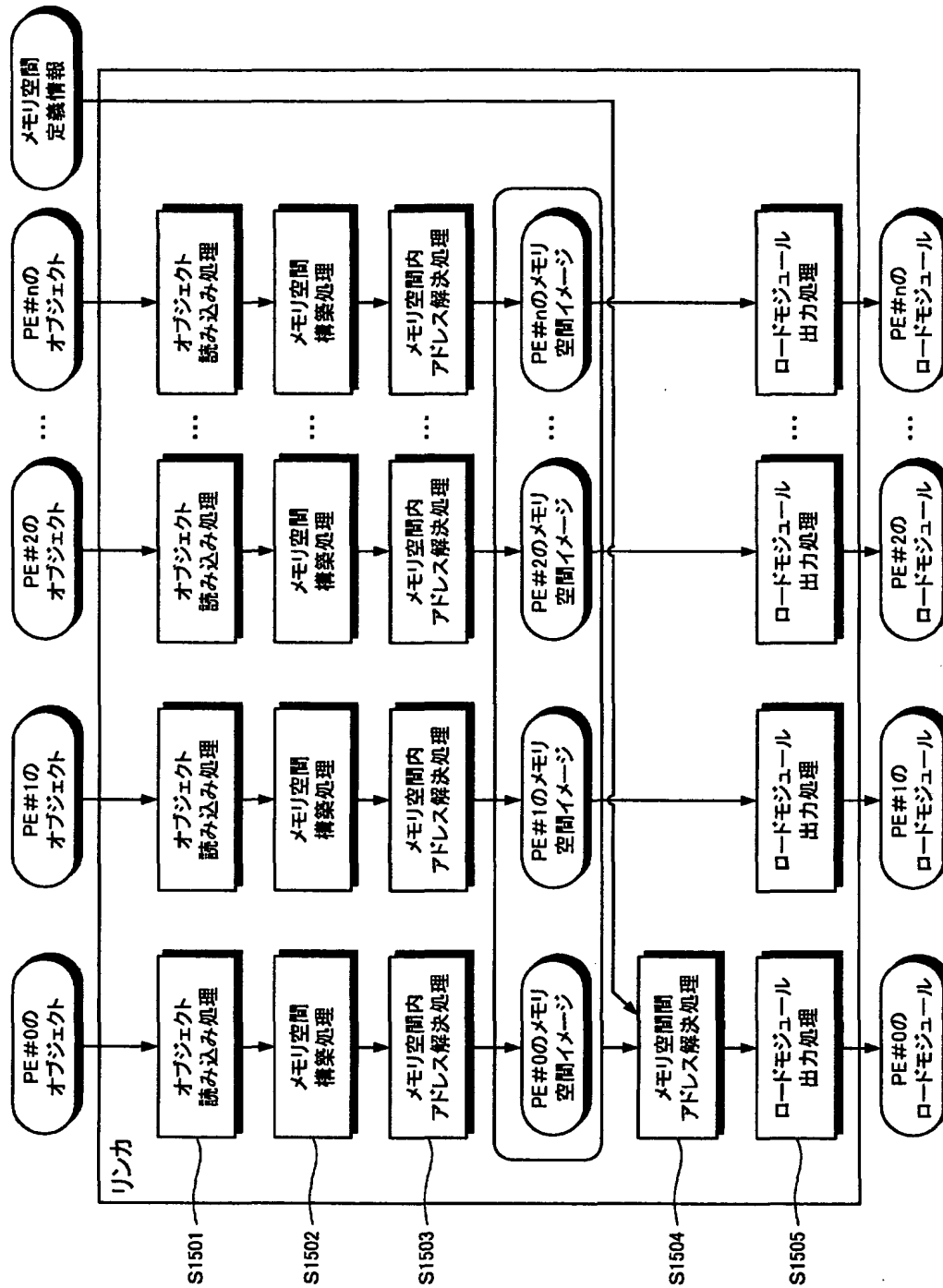
【図 1 4】

メモリ空間定義情報記憶部1306に保持されるメモリ空間定義情報の内容を模式的に示す説明図

PE識別	領域名	開始アドレス	終了アドレス
PE#0	text area	0x0000	0x0fff
	data area	0x1000	0x1fff
	shared data area #0	0x2000	0x2fff
	shared data area #1	0x3000	0x3fff
PE#1	text area	0x0000	0x0fff
	data area	0x1000	0x1fff
	shared data area #1	0x2000	0x2fff

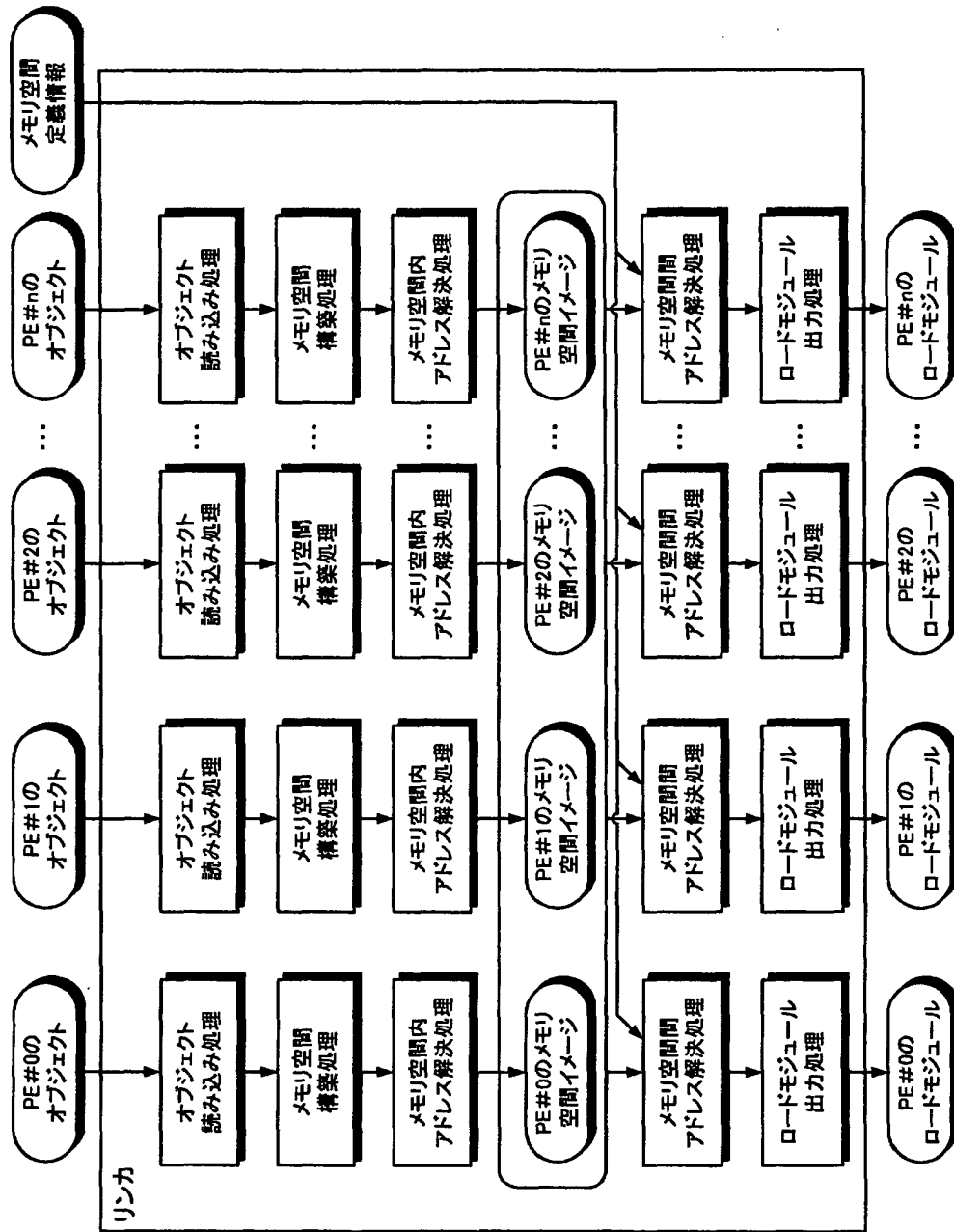
【図 15】

本発明の実施の形態にかかるロードモジュール生成装置における、図9および図10に示したプログラムのロードモジュール生成処理の手順を示すフローチャート



【図 16】

本発明の実施の形態にかかるロードモジュール生成装置における、図9および図10に示した
プログラムのロードモジュール生成処理の手順を示すフローチャート



【書類名】 要約書

【要約】

【課題】 分散共有メモリ型マルチプロセッサ方式を採用する計算機システムにおいて、SPMDでなくMPMDプログラミングにもとづくプログラムを動作させることでメモリの有効利用をはかるべく、現実に行実行可能な（すなわち、プログラム中の全シンボルのアドレスが解決された）当該プログラムのロードモジュールを生成するための言語処理系を提供すること。

【解決手段】 分散共有メモリ型マルチプロセッサ方式を前提とするMPMDプログラミングでは、たとえばPE # 0用のプログラムがPE # 1の共有メモリ上のデータを参照・変更することがあるが、当該データのアドレス（メモリ空間上での位置）はPEごとに異なっている。そこでメモリ空間間アドレス解決部1310により、PE # 1にとっての上記データのアドレスから、そのPE # 0にとってのアドレスを所定の計算式にもとづいて算出する。

【選択図】 図13

出 願 人 履 歴 情 報

識別番号 [0 0 0 0 0 5 2 2 3]

1. 変更年月日 1 9 9 6 年 3 月 2 6 日

[変更理由] 住所変更

住 所 神奈川県川崎市中原区上小田中 4 丁目 1 番 1 号

氏 名 富士通株式会社